

基于 GPU 的非相干消色散算法*

托乎提努尔^{1,3}, 张海龙^{1,2,3*}, 王杰^{1,3}, 冶鑫晨^{1,3}

(1. 中国科学院新疆天文台, 新疆乌鲁木齐, 830011; 2. 中国科学院射电天文重点实验室, 南京, 210008; 3. 国家天文科学数据中心, 北京, 100101)

摘要: 针对脉冲星信号实时消色散处理需求, 实现了基于 GPU 的非相干消色散算法。采用高性能并行计算方法对非相干消色散算法的多线程处理进行了深入研究, 提出了算法的并行化加速方案, 解决了消色散算法计算量大无法实时处理问题。分析算法的密集型计算部分, 高效利用 GPU 的层次存储结构, 提高了 GPU 资源利用率, 进而减少了计算时间, 显著提升了非相干消色散算法的计算性能。

关键词: 消色散; GPU; 并行计算; 实时处理

中图分类号: P111. 44 **文献标识码:** A **文章编号:**

0 引言

宇宙空间中的星际介质 (Inter-Stellar Medium, ISM)^[1] 包含大量电离气体云、中性尘埃粒子和自由电子等物质。脉冲星信号在宇宙空间中传播的时候会因为 ISM 色散的影响而降低速度, 高频的无线电波传播速度比低频快, 所以高频和低频电磁波到达射电望远镜的时间有一定延迟, 脉冲星信号因此出现能量分散而使脉冲轮廓加宽, 信噪比下降, 甚至会导致脉冲信号消失。

为了解决脉冲星信号色散问题, 天文学家研究了消除色散方法^[2-3]及高速消色散处理技术。利用多通道滤波器组^[4]对脉冲星信号进行通道划分, 生成多个窄带信号, 针对窄带信号进行时间延迟处理, 每个通道延迟时间可以通过色散公式计算, 最后将所有窄带通道进行累加, 获得高信噪比脉冲信号。

非相干消色散是脉冲星观测数据处理中最常用的色散处理方法, 具备实现简单、速度快、数据后期处理灵活等优势。在脉冲星搜寻中, 随着色散量 (Dispersion Measure, DM)、数据通道及采样数量的增加, 非相干消色散算法计算量迅速提高, 通用的计算平台难以实现脉冲星数据的实时处理。近年来, GPU^[5-6]的可编程能力及并行处理能力迅速提高, 应用范围也不断扩展, 在 CPU+GPU 混合计算系统中, GPU 的加入大大提高了整个系统的数据处理能力。高性能 GPU 集群系统可提供强大的计算资源, 能够满足海量天文数据的实时处理需求, 从而解决脉冲星消色散算法计算量巨大无法实时处理的问题。

1 非相干消色散

非相干消色散算法根据色散公式计算每个通道的延迟时间, 然后加上各个通道的延迟, 并把所有通道叠加在一起, 即可消除数据的色散影响。非相干消色散原理如图1所示。

非相干消色散采用多通道滤波器组来实现, 消色散处理过程主要包括: (1) 通道划分, 使用滤波器组把观测的天文信号总带宽分成若干个相互独立的狭窄通道; (2) 补充时间延迟, 根据色散公式计算每个通道的时间延迟, 按延迟进行通道平移, 将各个窄带通道的脉冲信号

* 基金项目: 国家自然科学基金(11873082, 11803080); 国家重点研发计划(2018YFA0404704); 中国科学院青年创新促进会: “西部之光”(2019-XBQNXZ-B-018); 国家天文科学数据中心, 中科院科学数据中心体系资助。

收稿日期: 2020-00-00; 修订日期: 2020-00-00

作者简介: 托乎提努尔, 男, 工程师. 研究方向: 数字终端与 GPU 并行计算技术. Email: nuer@xao.ac.cn

通讯作者: 张海龙, 男, 正高级工程师. 研究方向: 数据密集型研究. Email: zhanghailong@xao.ac.cn

在同一时刻对齐；(3) 通道累加，将所有通道时间序列叠在一起。

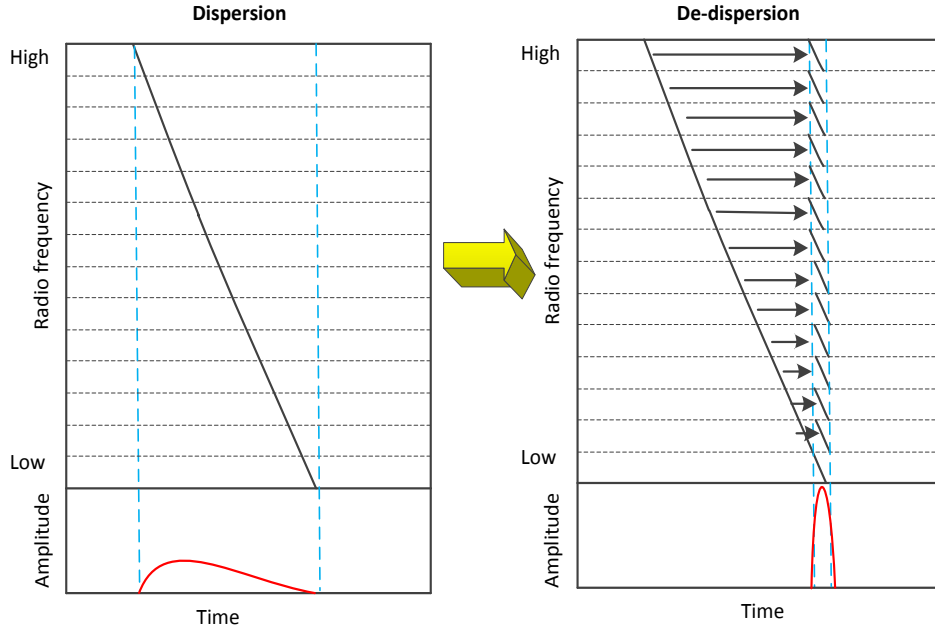


图1 非相干消色散原理

Fig 1. The principle of incoherent de-dispersion

图1说明了非相干消色散的处理过程。图中左半部分未进行消色散，右半部分是消色散处理后的结果。从图中可以看出，消色散前，通道累加之后脉冲宽度被展宽，输出信号信噪比下降，消色散之后可以得到信噪比大幅提高的脉冲星轮廓。

非相干消色散已被广泛应用于脉冲星、快速射电暴^[6]搜寻。非相干消色散方法处理后的脉冲星数据，其各个子通道内的色散延迟依旧存在，不能得到脉冲的真实轮廓，随着频谱通道数的增加，每个通道的带宽变小，带内的色散效应可相应减轻，低频信号 f_1 和高频信号 f_2 在星系际介质中的传播速度时间差为：

$$t_2 - t_1 = \frac{e^2}{2\pi cm} \times DM \times \left(\frac{1}{f_1^2} - \frac{1}{f_2^2} \right) \quad (1)$$

式中， c 为光在真空中的传播速度， e 为电子电荷， DM 为色散量， m 为电子质量。 DM 可表示为：

$$DM = \int_0^d n_e dl \approx n_e d \quad (2)$$

式中， n_e 为电子密度， d 为电磁波实际所经过的路径。

脉冲星消色散处理中，一个频率通道 f_{chan} 相对于参考通道 f_{ref} （通常是观测带宽中心频率）的时间延迟，可根据色散量公式（3）计算：

$$\Delta t = K_{DM} \times DM \times \left(\frac{1}{f_{ref}^2} - \frac{1}{f_{chan}^2} \right) \quad (3)$$

式中， K_{DM} 是色散量常数，DM 为观测脉冲星信号的色散量，其单位为 $\text{cm}^{-3} \text{pc}$ ，频率单位为 MHz。色散量常数为：

$$K_{DM} = \frac{e^2}{2\pi cm} = 4.148808 \times 10^3 \text{ MHz}^2 \text{ pc}^{-1} \text{ cm}^3 \text{ s} \quad (4)$$

在实际观测中，观测频率 f 往往远大于划分的通道带宽 Δf ，如果 $f \gg \Delta f$ 时，频率通道的延迟时间可写为：

$$t_{DM} = 8.3 \times 10^6 \text{ ms} \times DM \times \Delta f \times f^{-3} \quad (5)$$

式 (5) 说明，通道带宽和色散延迟时间成正比，为了得到更小的色散延迟，需要划分很细的窄带通道，尽量减少单通道信号带宽。

2 非相干消色散算法的 GPU 实现

GPU 是现代 PC 中常见的设备，专为执行复杂的数学和几何计算而设计的一种高度并行化、多线程、多核处理器，高速实现图形渲染。基于 GPU 的通用计算技术已经成为高性能并行计算领域的研究热点。GPU 由于具备多个核心，更适合计算密集型操作。GPU 的内核相当于 CPU 的多个线程，这些线程可以并行运行完成指定的计算任务，而且数值计算的速度远远优于 CPU。GPU 的线程结构如图 2 所示。

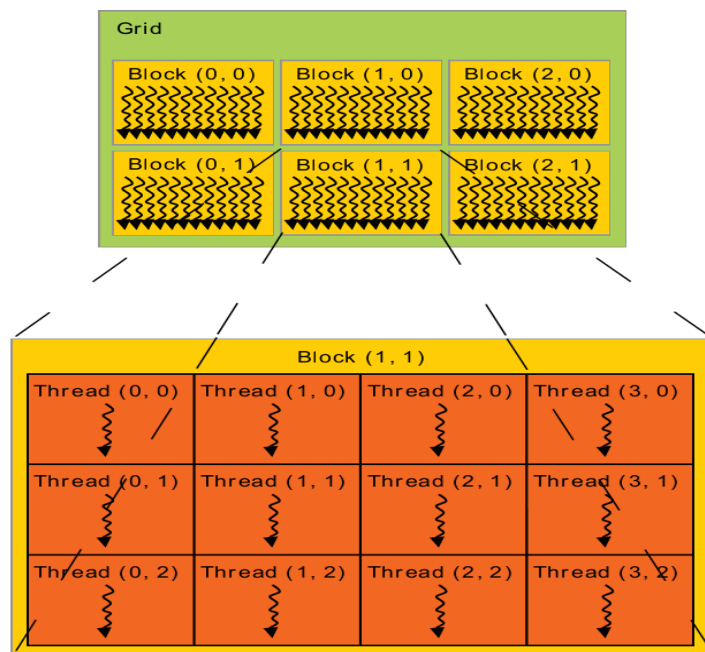


图 2 GPU 线程结构

Fig 2. GPU thread structure

GPU 的开发使用 CUDA^[8]程序, CUDA 是一种由 NVIDIA 推出的通用并行计算平台和编程模型, 使 GPU 能够解决复杂的计算问题。CUDA 提供了硬件的直接访问接口, 不依赖图形 API 接口来实现 GPU 访问, 在架构上采用了全新的计算体系结构来使用 GPU 提供的硬件资源。CUDA 采用标准 C 语言的扩展作为编程语言提供大量的高性能计算指令, 使我们能够在 GPU 的强大计算能力基础上实现效率更高的密集数据计算算法。

分析和研究非相干消色散算法结构, 把计算量大的任务部分映射到 GPU 的多线程进行处理。非相干消色散算法复杂度为 $O(N_d \times N_s \times N_c)$, 其中 N_d 为 DM 总数, N_c 为通道数, N_s 为采样数, 算法数学模型计算量较大, 但是通过 GPU 处理可得到很好的加速比。在 GPU 算法中, 对非相干消色散的色散量和时间维度上进行并行化, 频率通道累加采用了串行的处理方法, 即使用 GPU 的单线程实现了 N_c 个通道的相加计算。在整个消色散过程中, CPU 负责系统的初始化和输入数据读取, GPU 负责并行消色散处理, CPU 接收消色散后的时间序列并输出到数据文件。非相干消色散算法 CUDA 程序流程图如图 3 所示。

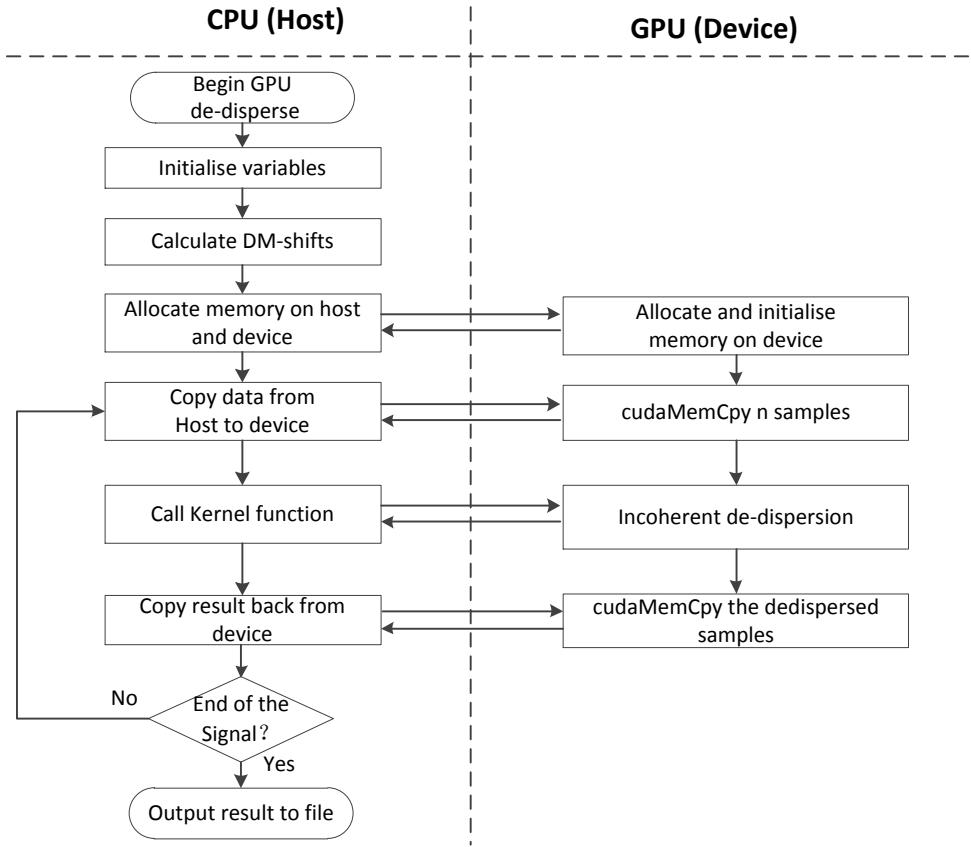


图 3 非相干消色散 CUDA 程序流程图

Fig 3. Flow chart of CUDA program for incoherent de-dispersion

首先在 CPU 内存中开辟缓冲区, 写入所需要处理的数据。缓冲区暂存的采样都存在一定的色散, 因此需要对各个频率通道进行位移和累加运算, 随着时间延迟增大, 通道位移也增大, 低频通道的位移量更大。为了减少 CPU 和 GPU 之间的频繁数据传输对算法计算性能的影响, 在 GPU 中也设置全局内存缓冲区, 尽量把大量数据一次性复制过去, 并且为写入和读取数据保留足够大存储空间。GPU kernel 函数执行过程中, 使用共享内存减少全局内存的延迟, 提高 kernel 函数运行速度。

为了提高算法的并行化加速, 将非相干消色散算法的计算分为两部分, 第一部分在 CPU

上执行。

$$t_{\text{chan}} = 4.15 \times 10^3 \times (f_1^{-2} - f_2^{-2}) \tag{6}$$

式中 f_1 、 f_2 的单位是 MHz， t_{chan} 的计算与 DM 值无关，使用 GPU 的常量内存存储。然后在 GPU 上执行计算的第二部分：

$$t_{\text{DM}} = \frac{t_{\text{chan}} \times DM}{t_{\text{samp}}} \tag{7}$$

式中 t_{samp} 为采样时间，单位是 ms。

根据非相干消色散算法特性，GPU kernel 函数中定义了两个缓冲区：共享内存和常量内存。其中，共享内存存储积分运算的结果，常量内存用于存储 DM_shift 。为了快速通道累加，使用共享内存，隐藏了全局内存的访问延迟。一个线程块里面的所有线程获取 DM 平移值，并且存储到共享内存。对于每一个 DM 值，在共享内存中进行积分运算，并把消色散处理结果写入到 GPU 的全局内存。

3 实验分析

本次实验平台使用了 Intel Xeon E5-1620、TITAN V、CUDA 10.0 及 Ubuntu 18.04。TITAN V 是一款 NVIDIA GeForce 系列高端 GPU，拥有 5120 个 CUDA 核，内存访问带宽为 652.8GB/s。

为了验证 GPU 并行算法性能，首先分别模拟生成了 32、64、128、256、512 及 1024 通道脉冲星数据，其中心频率为 420MHz，带宽为 6 MHz，采样间隔时间为 165 μs，脉冲星信号周期为 0.1s，然后将每一块数据单独读取并加载到 GPU 中进行处理。实验结果如表 1、表 2、表 3 及表 4 所示。

表 1 采样数为固定时非相干消色散处理时间（单位：s，采样：131072）

Tab.1 Incoherent de-dispersion processing time when the number of samples is fixed (time: s, samples: 131072)

number of DM	32 channel		64 channel		128 channel	
	CPU	TITAN V	CPU	TITAN V	CPU	TITAN V
1	0.0273	0.0078	0.0566	0.0148	0.1078	0.0289
2	0.0548	0.0085	0.1125	0.0154	0.2153	0.0296
5	0.1395	0.0102	0.2837	0.0172	0.5502	0.0324
10	0.2813	0.0132	0.5774	0.0204	1.1226	0.0357
20	0.5776	0.0189	1.2642	0.0265	2.3252	0.0434
40	1.4008	0.0314	2.7714	0.0393	5.3281	0.0599
80	3.0338	0.0574	6.0615	0.0658	11.9435	0.0962
160	6.1332	0.1193	13.0557	0.1273	26.7130	0.1636
320	13.8274	0.2556	29.5114	0.2740	56.2105	0.3399
640	33.9295	0.5106	65.7593	0.6378	134.9701	0.7307
1280	75.4416	1.0504	154.0248	1.3330	312.0603	1.7597
2560	158.7455	2.2094	329.0613	2.8760	668.0033	4.0038

5120	322.9089	4.3976	698.3460	6.0016	1389.3232	8.6268
------	----------	--------	----------	--------	-----------	--------

表 2 采样数为固定时非相干消色散处理时间（单位：s，采样：131072）
Tab. 2 Incoherent de-dispersion processing time when the number of samples is fixed
(time: s, samples: 131072)

number of DM	256 channel		512 channel		1024 channel	
	CPU	TITAN V	CPU	TITAN V	CPU	TITAN V
1	0.2103	0.0565	0.4204	0.1127	0.8289	0.2249
2	0.4308	0.0579	0.8371	0.1143	2.2732	0.2275
5	1.1644	0.0614	2.1853	0.1197	4.9677	0.2379
10	2.5279	0.0664	4.6357	0.1280	9.5110	0.2492
20	5.4891	0.0785	10.3499	0.1459	18.5704	0.2838
40	11.5288	0.1039	21.8757	0.1830	36.8282	0.3424
80	23.8050	0.1597	48.0757	0.2638	74.0204	0.4728
160	57.7672	0.2726	119.5114	0.4276	194.2584	0.7379
320	136.8641	0.5256	274.3375	0.7527	507.1785	1.2810
640	299.1812	1.0638	590.8383	1.4577	1161.6285	2.3831
1280	632.3722	2.5118	1238.2926	2.9597	2463.5231	4.6113
2560	1299.2245	6.8297	2592.5578	7.0263	5102.2142	9.3347
5120	2749.8164	19.221491	5533.0351	17.4777	10995.2138	20.4063

表 1、表 2 是在采样数为 131072（单通道采样），通道和 DM 数都变化的情况下 GPU 并行算法和 CPU 串行算法的非相干消色散处理消耗时间。从表中可以看出，当通道数量固定时，随着 DM 数的增加，GPU 和 CPU 的数据处理时间线性增加，GPU 的消色散处理时间远远少于 CPU 的计算时间；当 DM 数为固定时，随着数据通道数量的增加，CPU 和 GPU 处理需要的时间增大，但是 CPU 的计算时间比 GPU 大的多。

表 3 通道数为固定时非相干消色散处理时间（单位：s，通道：1024）
Tab. 3 Incoherent de-dispersion processing time when the number of channels is fixed
(time: s, channels: 131072)

number of DM	4096 sample		8192 sample		16384 sample		32768 sample	
	CPU	TITAN	CPU	TITAN	CPU	TITAN	CPU	TITAN
1	0.0259	0.0073	0.0519	0.0145	0.1037	0.0287	0.2072	0.0577
2	0.0518	0.0073	0.1036	0.0144	0.2075	0.0286	0.4147	0.0569
5	0.1307	0.0077	0.2624	0.0151	0.5251	0.0298	1.0499	0.0598
10	0.2656	0.0081	0.5379	0.0157	1.0764	0.0313	2.1585	0.0625
20	0.5430	0.0086	1.1102	0.0171	2.2238	0.0348	4.4429	0.0699
40	1.0930	0.0099	2.2470	0.0205	4.5026	0.0414	9.1970	0.0852
80	2.2684	0.0129	4.6959	0.0266	9.3686	0.0565	18.5134	0.1147
160	5.4016	0.0191	12.2362	0.0401	24.2389	0.0859	48.6114	0.1781
320	12.6597	0.0296	31.9656	0.0705	63.5571	0.1461	126.8976	0.3047
640	28.2764	0.0541	71.4992	0.1230	144.0788	0.2692	288.2967	0.5664
1280	61.5773	0.1020	150.4389	0.2408	299.1604	0.5263	614.7442	1.1099
2560	129.2098	0.2064	302.7521	0.4777	637.8879	1.0559	1280.9965	2.2027

5120	281.6858	0.4243	672.2270	0.9917	1374.9966	2.1815	2756.7142	4.5928
------	----------	--------	----------	--------	-----------	--------	-----------	--------

表 3 是当数据为 1024 通道时，采样和 DM 数都变化情况下的 GPU 并行算法和 CPU 串行算法的非相干消色散处理消耗时间。当采样数固定时，随着 DM 数的增加，GPU 和 CPU 的数据处理时间增加；当 DM 数为固定时，随着数据采样的增加，并行和串行算法执行需要更多的时间。但是，GPU 的色散处理时间少于 CPU 的计算时间，计算速度有几百倍的差距，从表中可以看出 GPU 大大缩短了非相干消色散的执行时间。

表 4 是在 DM 数为 5120，采样和通道数都变化时，GPU 并行算法和 CPU 串行算法的非相干消色散处理消耗时间。当采样固定时，随着数据通道的增加，GPU 和 CPU 的数据处理时间也增加；当通道数固定时，随着数据采样的增加，并行和串行算法执行需要更多的时间。在通道数或采样数变化的情况下，GPU 的运算时间均小于 CPU，且消耗时间差距明显。

表 4 DM 数为固定时非相干消色散处理时间（单位：s，DM：5120）

Tab. 4 Incoherent de-dispersion processing time when the number of DMs is fixed (time: s, DMs: 131072)

Channel	Sample	4096	8192	16384	32768	65536
32	CPU	7.4468	15.0353	30.2430	60.6416	133.7846
	TITAN V	0.1002	0.2003	0.4215	0.8864	2.0978
64	CPU	14.5728	29.8505	59.7925	123.8967	330.8217
	TITAN V	0.1127	0.2164	0.4667	1.2664	2.8659
128	CPU	28.6676	57.8297	118.1305	316.7424	703.6475
	TITAN V	0.1307	0.2665	0.7702	1.9095	4.0631
256	CPU	53.8793	111.2041	301.5122	694.0670	1386.7769
	TITAN V	0.1519	0.4544	1.1833	3.4037	8.927236
512	CPU	104.1580	282.3113	684.1435	1361.1676	2718.7850
	TITAN V	0.2411	0.6064	1.4260	3.3107	7.3363
1024	CPU	281.2189	684.9610	1375.0873	2744.7187	5494.2190
	TITAN V	0.4230	1.0071	2.1739	4.6273	9.8898

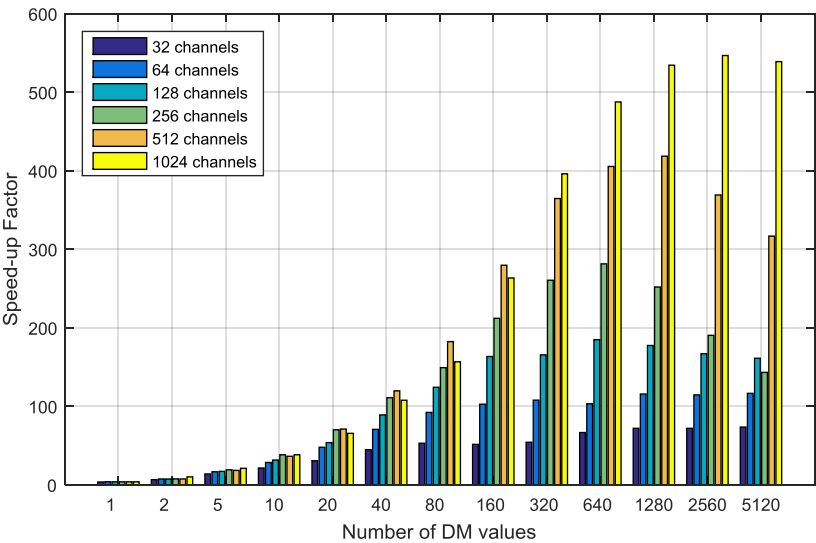


图 4 采样数为 131072 时 GPU 算法加速比

Fig 4. GPU algorithm speed-up when the number of samples is 131072

和CPU比较,GPU非相干消色散的数据能力达到几百倍的加速比,具有显著的加速优势。GPU并行算法的加速比性能如图4、图5及图6所示。

图4是在采样数为131072,通道和DM数均变化的情况下GPU并行算法的加速比。从图中可以看到,随着DM数的增加,并行算法的加速比提高。当DM个数为2560时,TITAN V的加速比最高(即达到CPU速度的538倍),然后加速比开始下降。如图5所示,通道数量越多,GPU算法的加速比越大,得到的加速性能越好。当通道数为1024时,加速比最高;当通道数为32时加速比最小。

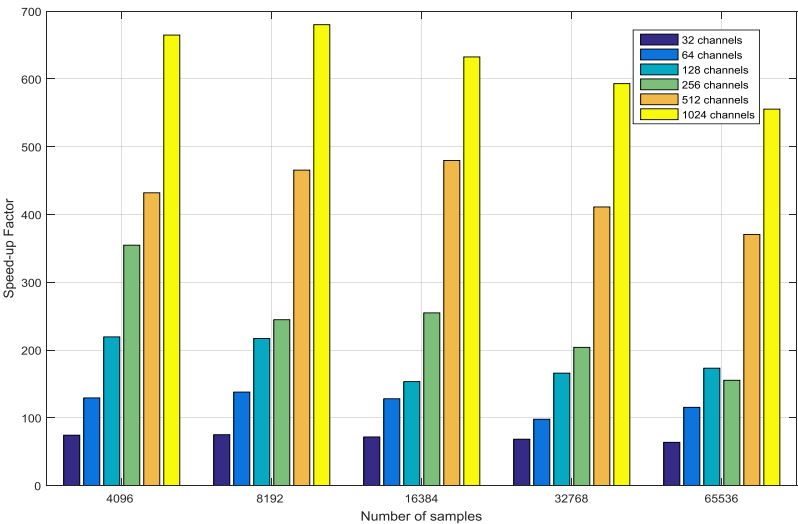


图5 DM数为5120时GPU算法加速比

Fig 5. GPU algorithm speed-up when the number of DMs is 5120

图5显示的是当DM数为5120时,在通道数量和采样数都变化的情况下GPU并行算法的加速比。从图中可以看到,随着采样的增加,并行算法的加速有所下降,但算法加速高达550多倍。当通道数为1024时,TITAN V达到最高的加速比。当通道数为1024、采样为8192时,GPU并行算法是CPU算法速度的780多倍。

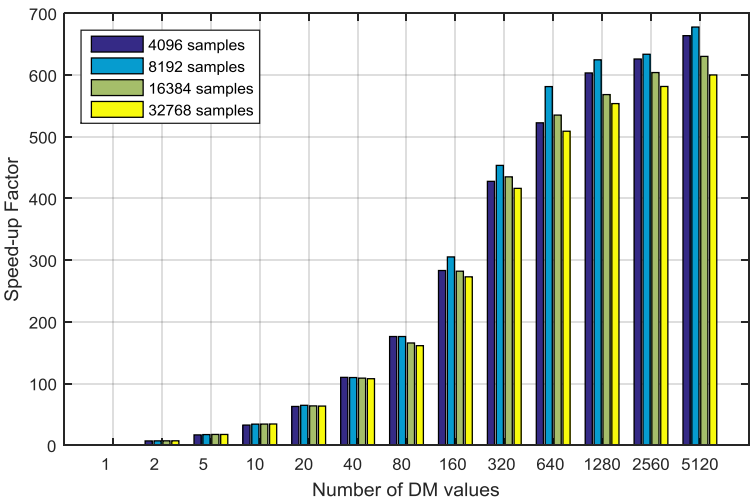


图6 通道数为1024时GPU算法加速比

Fig 6. GPU algorithm speed-up when the number of channels is 131072

图 6 是在通道数为 1024，采样和 DM 数都变化的情况下 GPU 并行算法的加速比。从图中可以看到，DM 个数对 GPU 非相干消色散算法加速比影响最大。随着 DM 数的增加，GPU 并行算法的加速比迅速提高。如图 6 所示，DM 个数越大，计算消耗时间越多，当 DM 数为 5120 时，加速比最大。

实验结果表明，采样数、通道数及 DM 值个数都是影响 GPU 算法加速性能的关键因素。非相干算法执行中 GPU 展示了强大的并行化处理优势，节省了大量重复计算时间，提升了算法的计算效率。

4 总结

本文研究基于GPU的非相干消色散算法，提出了相关算法的GPU并行化加速方案。分析了算法的密集型计算部分、研究了GPU多线程任务分配、管理及寄存器层次的优化方法，提高了GPU资源利用率，显著提升了算法的计算性能。在GPU算法的实现过程中，深入分析了影响并行算法性能的主要因素，优化了CUDA程序，大大减少了算法执行消耗时间，GPU消色散算法加速比接近700倍，解决了算法在CPU上计算量巨大无法实时处理的问题。通过实验分析了算法的数据处理消耗时间和加速比，验证了并行算法性能。

参考文献：

- [1] Spitzer Jr L. Physical processes in the interstellar medium[M]. John Wiley & Sons, 2008.
- [2] Barsdell B R, Bailes M, Barnes D G, et al. Accelerating incoherent dedispersion[J]. Monthly Notices of the Royal Astronomical Society, 2012, 422(1): 379-392.
- [3] 黄玉祥,汪敏,郝龙飞,李志玄,徐永华.脉冲星信号相干消色散与非相干消色散的比较研究[J].天文研究与技术(Huang Yuxiang, Wang Min, Hao Longfei, Li Zhixuan, Xu Yonghua. Comparative Study between the Coherent De-dispersion and the Incoherent De-dispersion of Pulsar Signal. Astronomical Research & Technology),2019,16(01):16-24.
- [4] 托乎提努尔,张海龙,王杰.基于 CUDA 的射电天文多相滤波器组设计[J].天文研究与技术(Tohtonur, Zhang Hailong, Wang Jie. A Design of Polyphase Filter Bank for Radio Astronomy Based on CUDA. Astronomical Research & Technology),2017,14(01):117-123.
- [5] 苏统华,李东,李松泽. CUDA 并行程序设计 GPU 编程指南[M]. 机械工业出版社(Su Tonghua, Li Dong, Li Songze. CUDA Programming: A Developer's Guide to Parallel Computing with GPUs. China Machine Press), 2014.
- [6] Rob Farber. 高性能 CUDA 应用设计与开发[M]. 机械工业出版社(Rob Farber. CUDA Application Design and Development. China Machine Press), 2013.
- [7] Petroff E, Oostrum L C, Stappers B W, et al. A fast radio burst with a low dispersion measure[J]. Monthly Notices of the Royal Astronomical Society, 2019, 482(3): 3109-3115.
- [8] Cheng J, Grossman M, McKercher T. Professional CUDA C programming[M]. John Wiley & Sons, 2014.

Incoherent Dedispersion Algorithm Based on GPU

Toktonur^{1,3}, Zhang Hailong^{1,2,3*}, Wang Jie^{1,3}, Ye Xincheng^{1,3}

(1.Xinjiang Astronomical Observatory, Chinese Academy of Sciences, Urumqi 830011,China; 2. Key Laboratory of Radio Astronomy, Chinese Academy of Sciences, Nanjing 210008,China; 3. National Astronomical Data Center, Beijing 100101, China)

Abstract: In order to meet the requirements of real-time de-dispersion processing of pulsar signals, a GPU based incoherent de-dispersion algorithm is implemented. We use high-performance parallel computing methods to conduct in-depth research on the incoherent de-dispersion algorithm for multi-thread processing, and propose a parallel acceleration scheme for the algorithm, which solves the problem that the de-dispersion algorithm has a large amount of calculation and cannot be processed in real time. We analyze the intensive computing parts of the algorithm, efficiently use the hierarchical storage structure of the GPU, improve the utilization rate of GPU computing resources, thereby reducing the calculation time, and significantly improving the calculation performance of the incoherent de-dispersion algorithm.

Key words: De-dispersion; GPU; Parallel Computing; Real-time Processing